IEICE TRANS. INF. & SYST., VOL.E99–D, NO.10 OCTOBER 2016

PAPER

Shilling Attack Detection in Recommender Systems via Selecting Patterns Analysis

Wentao LI[†], **Min GAO**[†,††a)], **Hua LI**[†,†††], **Jun ZENG**[†,††], **Qingyu XIONG**[†,††], ***Nonmembers***, ***and*** **Sachio HIROKAWA**[††††], ***Member***

SUMMARY Collaborative filtering (CF) has been widely used in recommender systems to generate personalized recommendations. However, recommender systems using CF are vulnerable to shilling attacks, in which attackers inject fake profiles to manipulate recommendation results. Thus, shilling attacks pose a threat to the credibility of recommender systems. Previous studies mainly derive features from characteristics of item ratings in user profiles to detect attackers, but the methods suffer from low accuracy when attackers adopt new rating patterns. To overcome this drawback, we derive features from properties of item popularity in user profiles, which are determined by users' different selecting patterns. This feature extraction method is based on the prior knowledge that attackers select items to rate with man-made rules while normal users do this according to their inner preferences. Then, machine learning classification approaches are exploited to make use of these features to detect and remove attackers. Experiment results on the MovieLens dataset and Amazon review dataset show that our proposed method improves detection performance. In addition, the results justify the practical value of features derived from selecting patterns.

key words: *feature extraction, popularity, selecting patterns, recommender systems, shilling attacks*

1. Introduction

Collaborative filtering (CF) is a technique widely used in recommender systems [1], [2]. Recommender systems using CF bring huge profits for industries by creating excellent recommendation results for users [3]. The filtering process of CF is based on user profiles [4], [5], and therefore it fails to work well when encountering shilling attacks [6], [7], in which attackers inject fake profiles to manipulate recommendation results made by CF [8]. As a result, attackers can insert spam ratings or reviews to mislead the normal uses in recommender systems [9]–[11].

According to the purpose of attacks, shilling attacks can be classified as push attacks and nuke attacks. The former tries to make a target item easier to be recommended, while the latter does the contrary [12], [13]. Shilling attacks reduce the prediction accuracy of recommender systems [12]. As a result, the recommended items for users do not match their preferences, which affect users' satisfaction. Prior research shows that the target item can be pushed to the top of a recommendation list by inserting one percent fake profiles [14]. Therefore, how to detect shilling attacks is of great significance to the robustness of recommender systems [13].

We define injected profiles as *attack profiles* [13], then the task of shilling attack detection is to find attack profiles [15]. In recent years, there have been a lot of studies on shilling attack detection. In general, there are three categories of methods: supervised, unsupervised, and semi-supervised methods [15]–[17]. The common aspect of these methods is that the features they used are mostly derived from user rating patterns, namely extracting attributes from the ratings assigned to items in user profiles [13]. Two main problems exist with these features: (1) The ratings given by some normal users look similar to those of attackers, which easily leads to misjudgment of these normal users; (2) These features, such as RDMA (Rating Deviation from Mean Agreement) [16] or DegSim (Degree of Similarity with Top Neighbors) [16], are extracted based on a certain kind of attack. Hence, detection methods based on these features do not fit various changes of shilling attacks in the actual systems.

To solve these problems, the paper makes the empirical analysis of users' popularity characteristics caused by different selecting patterns between normal users and attackers. Then features are derived from the popularity distribution of rated items in user profiles. We denote the popularity of an item as the number of ratings it has. The principle behind these features is that attackers have limited knowledge about the system, hence, attackers select items to rate with man-made rules while normal users do that in accordance with their inner preferences. Because the popularity of items in the systems follows the power-law distribution, popularity distribution of rated items in attackers' profiles will differ from those of normal users caused by different selecting strategies. After extracting features, machine learning algorithms are exploited to get our attacker detection method (Pop-SAD). The results on the MovieLens dataset show that Pop-SAD achieves better detection performance than methods using features derived from rating patterns. Moreover, experiments conducted on the Amazon review dataset demonstrate the practical aspect for shilling attack

Manuscript received December 18, 2015.
Manuscript revised May 11, 2016.
Manuscript publicized June 27, 2016.

[†]The authors are with the Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 400044, China.
[††]The authors are with the School of Software Engineering, Chongqing University, Chongqing 400044, China.
[†††]The author is with the College of Computer Science, Chongqing University, Chongqing 400044, China.
[††††]The author is with the Research Institute for Information Technology, Kyushu University, Fukuoka-shi, 812–8581 Japan.
a) E-mail: gaomin@cqu.edu.cn

DOI: 10.1587/transinf.2015EDP7500

detection of our proposed method.

The rest of this paper is organized as follows: In the next section, we summarize some previous works. In Sect. 3, we present some empirical analysis of popularity characteristics caused by selecting patterns. Section 4 presents the proposed detection method (Pop-SAD). In Sect. 5, we report experimental results made on the MovieLens and Amazon review datasets against some existing methods. Finally, we conclude in Sect. 6.

2. Related Work

In this section, we first illustrate some common attack models and their characteristics. Then, we describe two features extracted from user rating patterns. Finally, we review several notable shilling attack detection algorithms.

2.1 Attack Models

The user profile refers to a collection of user ratings to all items [6]. In order to behave like normal users, the attackers use attack models to create attack profiles based on their knowledge about the recommender systems, such as rating database and item popularity [18]. The general form of an attack profile is shown in Table 1 [15].

As shown in Table 1, an attack profile consists of a $|I|$-dimensional vector of ratings, where $|I| = (s+f+nl+t)$, s, f, nl, and t denote the number of items in sets I_S, I_F, I_N, and I_t respectively [18]. We describe the details of these four sets as follows.

(1) I_S is the set of selected items based on specific needs of the attacker. Items in I_S are given rating values by function α. I_S is not a necessary for some attack models.

(2) I_F is the set of filler items and is used to decrease the sparseness of the attack profiles in order to disguise attackers. Items in I_F are given rating values by function β and are usually chosen randomly.

(3) I_N is the set of unrated items.

(4) I_t is the set of target items selected to be attacked. For each attack profile, there is usually a single target item, i_t, in it. The rating given to it will be the maximum for push attack or minimum for nuke attack as determined by the function γ.

Table 1 General form of an attack profile

I_S	I_F	I_N	I_t
$I_{S,1} \cdots I_{S,s}$	$I_{F,1} \cdots I_{F,f}$	$I_{N,1} \cdots I_{N,nl}$	I_t
$\alpha(I_{S,1}) \cdots \alpha(I_{S,s})$	$\beta(I_{F,1}) \cdots \beta(I_{F,f})$	Null $\cdots$ Null	$\gamma(I_t)$

Let $s/|I|$ be *selected size*, which is the ratio between the number of items I_S and the length of a attack profile. Let $f/|I|$ be *filler size*, which is the ratio between the number of items I_F and the length of a attack profile. We also define *attack size* as the ratio between the number of inserted attack profiles and the whole normal profiles in the rating database.

The *attack model* is used to create attack profiles [15], [18], so it can be defined by the strategies for selecting items in sets I_S, I_F and the rating functions α, β, and γ for giving rating values to items [18]. Several common attack models are summarized in Table 2. We define the strategies for selecting items as selecting patterns and the strategies for giving rating values to items as rating patterns.

In Table 2, we refer to μ and σ as the mean and standard deviation of ratings for all items, μ_i and σ_i as the mean and standard deviation of ratings for item i, and r_{max} and r_{min} are the maximum and minimum values in the rating database [13], [19].

Among these models, the random attack model is a naive attack in which filler items are those randomly chosen using random values. The average attack model is a more sophisticated attack model and items in the filler set are randomly chosen using the average rating value. The bandwagon attack model is the extensions to the basic attack models. Besides filler items, the selected items in bandwagon attacks are those frequently rated with a high rating. The average of popular items attack (Aop attack) model is an obfuscated version of the average attack model in which items are chosen with equal probability from the top x% of most popular items rather than the entire items.

The hybrid attack model, which combines various types of common attack models together, is very likely to be adopted by shilling attackers to make their attacks more difficult to detect [15]. Therefore, the attack models described in our paper represent those which are widely adopted in academia and industry [20].

2.2 Features Derived from Rating Patterns

A feature extraction method is used because the dimension of a user profile is so high that computing directly on a profile becomes impossible [21]. Existing shilling attack detection methods mostly derived features from rating values in

Table 2 Summary of four common shilling attack models

Attack model	I_S	I_F	I_t
Random attack	$\emptyset$	A set of randomly chosen items drawn from $I - I_t$, and $\forall\, i \in I_F, \beta(i) \sim N(\mu;\sigma)$	$\gamma(i_t) = r_{max}$ for push attacks and $\gamma(i_t) = r_{min}$ for nuke attacks
Average attack	$\emptyset$	A set of randomly chosen items drawn from $I - I_t$, and $\forall\, i \in I_F, \beta(i) \sim N(\mu_i;\sigma_i)$	$\gamma(i_t) = r_{max}$ for push attacks and $\gamma(i_t) = r_{min}$ for nuke attacks
Bandwagon attack	A set of some most popular items and $\forall\, i \in I_S, \alpha(i)=r_{max}$	A set of randomly chosen items drawn from $I - I_t - I_S$, and $\forall\, i \in I_F, \beta(i) \sim N(\mu;\sigma)$	$\gamma(i_t) = r_{max}$ for push attacks and $\gamma(i_t) = r_{min}$ for nuke attacks
AoP attack	$\emptyset$	A set of randomly chosen filler items drawn from the top x% of most popular items in $I - I_t$ where x% is a pre-specified decimal value and $\forall\, i \in I_F, \beta(i) \sim N(\mu;\sigma)$	$\gamma(i_t) = r_{max}$ for push attacks and $\gamma(i_t) = r_{min}$ for nuke attacks

the user profiles [6]. As attackers gain limited knowledge about the systems, their ratings given to items are different from those of normal users [6]. Among these features, DegSim and RDMA are the most popular ones and are calculated according to the formulas (1) and (2) [6], [21], [22].

- Degree of similarity with Top Neighbors (DegSim)

$$Degsim_u = \frac{\sum_{v=1}^{k} W_{u,v}}{k} \quad (1)$$

where $W_{u,v}$ is the similarity between user u and user v. The mean similarity value of user u's top k most similar users is denoted as $DegSim_u$.

- Rating Deviation from Mean Agreement (RDMA)

$$RDMA_u = \frac{\sum_{i=0}^{N_u} \frac{|r_{u,i} - \bar{r}_i|}{NR_i}}{N_u} \quad (2)$$

where N_u is the number of items user u rated, $r_{u,i}$ is the rating of user u given to item i, $\bar{r}_i$ is the mean rating value that item i got in the rating database, NR_i is the number of ratings that item i has been given in the system.

2.3 Shilling Attack Detection Methods

In 2004, shilling attacks were noticed by researchers who studied the effect of attack models against recommender systems [7]. Since then, numerous efforts have been made to develop methods to detect shilling attacks. According to the amount of labels needed to train a model, these methods can be divided into three categories.

- Attack detection based on supervised learning
  In this type, a classifier is trained with some detection features to classify two types of users. For example, Chirita et al. [16] proposed two features, DegSim and RDMA, to detect attackers; Williams et al. [9], [22] systematically defined detection features for shilling attacks, and then proposed a decision tree algorithm for shilling attack detection.
- Attack detection based on unsupervised learning
  Unsupervised learning methods get the detectors without labels of users. Mehta et al. [17] first put forward the unsupervised learning based detector, PCA-SelectUsers. Zhang et al. [23] put forward a large component searching (LC) algorithm to find the most associated sub-matrices in a user-user similarity matrix to detect shilling attackers.
- Attack detection based on semi-supervised learning
  In the real situation, there exist few labeled data, thus semi-supervised learning method is used to detect attackers. Cao et al. [15] firstly proposed a semi-supervised learning framework called Semi-SAD for shilling attack detection and EM algorithm is used to estimate the parameters, then they proposed HySAD, in which feature automatic selection function was added, to detect shilling attacks [20].

Although the above-mentioned algorithms perform well in shilling attack detection, they just rely on rating patterns of users to find attackers. However, attackers can adopt new strategies for giving rating values to evade detection. In this paper, we derive features from selecting patterns rather than rating patterns. We hope our proposed features can discriminate attack users from the popularity perspective and increase the cost of attacks.

3. Popularity Analysis of User Profiles

Unlike normal users, attackers aim at promoting or suppressing the target items [7], [13]. Thus, the selecting patterns of attackers differ from those of normal users. While traditional research studied the characteristics related to rating patterns, we study the popularity characteristics of users.

Intuitively, because items in the recommender systems have different popularity values, the popularity distributions of rated items in user profiles will be different if users select items to rate with different mechanisms. In order to verify this intuition, we first look at the distribution characteristics of item popularity in a recommender system, and then show the popularity characteristics of rated items in user profiles.

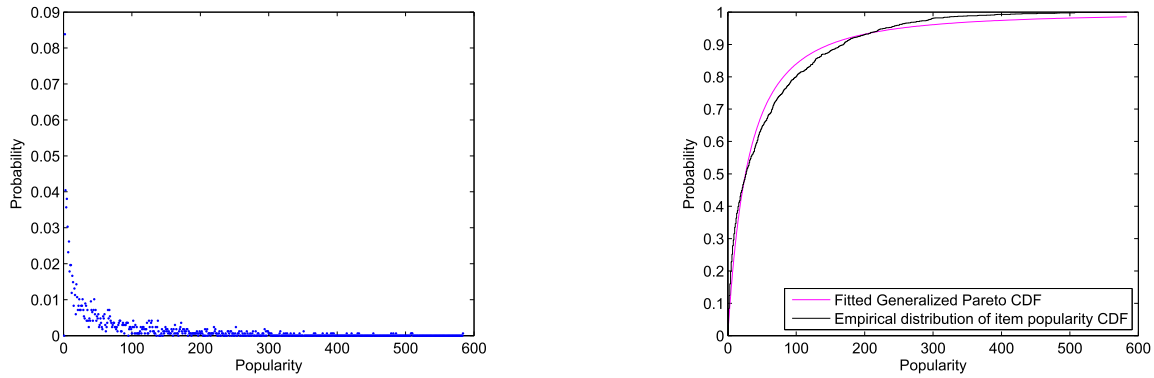
3.1 Property of Item Popularity Distribution

The item popularity refers to the number of ratings an item has been given [24], [25]. The high popularity of an item means the item is quite popular among users. Our purpose is to find the distribution that item popularity values follow. Towards this goal, we use the MovieLens 100K dataset[†] issued by GroupLens as an example to cover the property of distribution. MovieLens 100K is a rating set crawled from the MovieLens web site (http://movielens.org). This dataset consists of the ratings from 943 users on 1682 items, with a rating frequency not less than 20 for each user. The paper assumes that users in the original dataset are normal users and the inserting users are attackers.

Figure 1 (a) shows the cumulative distribution function of items' popularity in the system. We find from the figure that a large fraction of the items' popularity locates at the lower level, which means only a few items selected by a lot of people. The phenomenon is known as the long tail effect [26], [27]. This kind of effect is also known as the power law distribution. In order to fit the item popularity in a recommender system, the generalized Pareto distribution is selected as a theoretical model to fit them [28]. The fitting result is shown in Fig. 1 (b).

It can be seen from the fitting chart that item popularity distribution complies with Pareto distribution to some extent. More proof details can be found in [24]. Thus, one unbalance can be foreseen: the probability of each item selected by customers is not equal, and a few of items are much popular than others. This leads to the conclusion that if users choose items to rate with the different mechanisms,

[†]http://grouplens.org/datasets/movielens/



(a) (b)

Fig. 1 Property of item popularity distribution, (a) distribution of item popularity, (b) fitting result.

then the item popularity in their profiles will be different.

3.2 Popularity Characteristics of User Profiles

Normal users and attackers select items to rate with different mechanism: Normal users select items to rate with inner preference while attackers do that with rules according to the attack models. While item popularity values follow power-law distribution, this leads to the difference of popularity characteristics of rated items in user profiles due to various selecting mechanisms.

In the previous section, the user profile refers to a collection of user ratings to all items. Before we check the popularity characteristics of user profiles, we give some definitions below.

Definition 1: The popularity profile refers to a set of item popularity values of rated items.

We denote the popularity profile of user u as $PP_u = (d_{u,1}, d_{u,2}, \ldots, d_{u,N_u})$, where N_u is the number of items user u has rated. du_i is item i's popularity value if user u gives a rating to item i.

For example, if there are three items in the system which are denoted by i_1, i_2, and i_3, with popularity values denoted by d_1, d_2, and d_3. If user u gives ratings only to i_1 and i_3, the popularity profile of user u is (d_1, d_3). Item i_2's popularity is not included because each element of popularity profile is the popularity of rated items.

Definition 2: The popularity distribution of a user refers to the probability distribution of item popularity values in the user's popularity profile PP_u.

Each element in this distribution is the probability of elements in PP_u whose popularity values equal to a certain value. The probability can be calculated as $p_{u,i} = N_{d=i}/N_u$. Where N_u is the number of elements in user u's popularity profile, $N_{d=i}$ is the number of elements with popularity equal to i. Note that we do not take popularity values equal to zero into account. Then popularity distribution of user u is denoted as $D_u = (p_{u,1}, p_{u,2}, \ldots, p_{u,d_{max}})$, where d_{max} is the maximum value of popularity in the system.

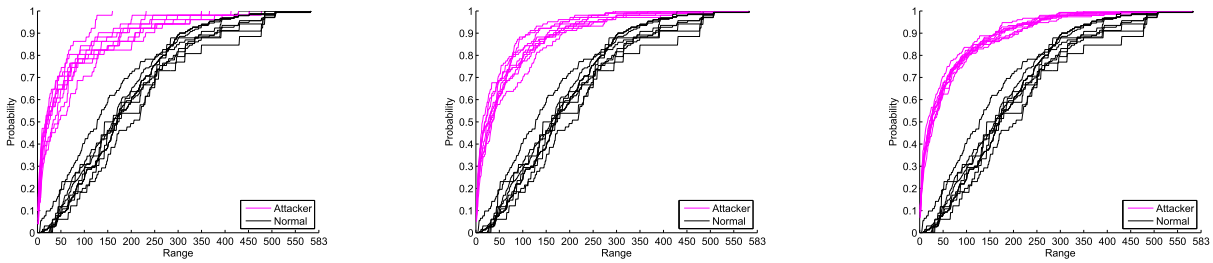
As can be seen from the typical shilling attack models in Sect. 2.1, there are three kinds of attacks when only selecting patterns are considered [6], [7]: 1) Attackers who randomly choose items to rate from the whole item set, such as random attacks and average attacks; 2) Attackers who randomly choose items to rate from the top-x% popular item set, such as average over popular attacks (Aop attacks); 3) Attackers who randomly choose items to rate both from the whole item set and from certain kind of item set, such as bandwagon attacks. Other attackers can be included in these three categories. Attacker profiles which are generated by random, Aop and bandwagon attack models are analyzed to get popularity characteristics of user profiles.

We set attack size equals to 10%, and filler size changes from 3%, 6% to 9% for random attacks; top-x% changes from 20%, 40% to 60% for AoP attacks; ratio between selected size and filler size, changes from 0.5, 1 to 1.5 for bandwagon attacks. We randomly choose ten normal user profiles and ten attack profiles to report results. Then the comparison of two user groups' cumulative probability function of popularity distribution is shown in Figs. 2-4.

We notice from all three charts that popularity of rated items in normal users located evenly as the value of popularity increases. It can also be noticed from Fig. 2 that if attackers selected items randomly from the whole item set, their popularity values mainly located in the smaller value area. The reason is that item popularity values follow the power-law distribution, therefore, the items with lower popularity are more likely to be hit if selected blindly.

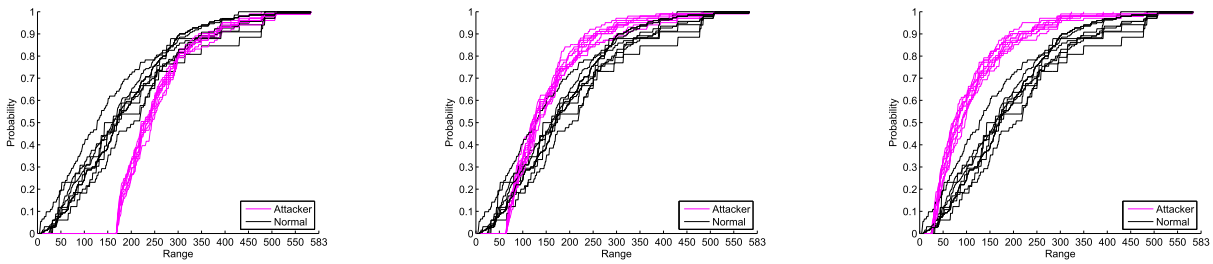
In addition, as seen from Fig. 3, if attackers selected items randomly from the popular item set, then their popularity mainly located in the larger value area. The reason is that all selected items have large popularity values. Now we turn to the Fig. 4, it can be found that if attackers selected items from both item set, a clear turning point would be found in the cumulative probability function.

To summarize, popularity distributions of users change as selecting patterns change. Though attackers can adopt new selecting patterns, their knowledge about the rating database of a recommender system is limited, therefore, their selecting behavior still show difference. Moreover, the purpose of attackers is to change the prediction of target items, so attackers will select target items while normal



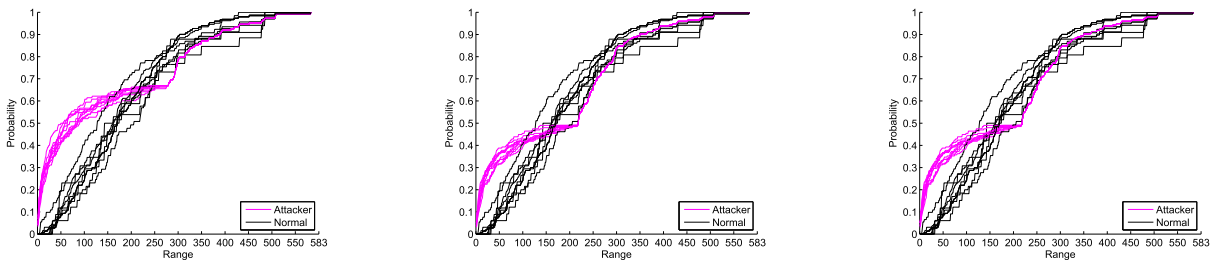
(a) (b) (c)

Fig. 2 Contrast of cumulative probability function (random attacks), (a) filler size = 3%, (b) filler size = 6%, (c) filler size = 9%.



(a) (b) (c)

Fig. 3 Contrast of cumulative probability function (Aop attacks), (a) top-x% = 20%, (b) top-x% = 40%, (c) top-x% = 60%.



(a) (b) (c)

Fig. 4 Contrast of cumulative probability function (bandwagon attacks), (a) ratio = 0.5, (b) ratio = 1, (c) ratio = 1.5.

users will not. As a result, popularity will always show the difference if they want to shill the target items. Consequently, we can use popularity distributions of rated items in user profiles to extract features.

4. The Proposed Method

4.1 Overview

Popularity Characteristics of user profiles are analyzed in the former section. We find that the popularity distributions of rated items in normal user profiles and attack profiles are different due to different selecting patterns. In this section, we propose a shilling attack detection algorithm named Pop-SAD. The framework in Fig. 5 is given as a description of Pop-SAD.

The left side of the framework is data preprocessing stage. The popularity of an item can be obtained through

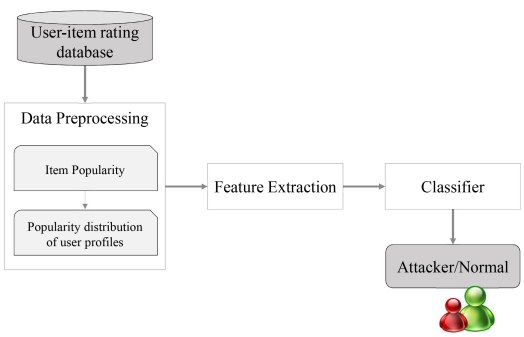


Fig. 5 Framework of the proposed method

counting ratings the item has got in the rating database. Then the popularity profile of each user is formed according to definition 1. The middle side of the framework is the feature extracting stage. Through accumulating probabil-

Table 3 Bucket the range of the popularity distribution

Interval	1	2	3	4	5	6
X-Values	1-100	101-200	201-300	301-400	401-500	501-583

ity values over several intervals, each user is represented by these probability values. The right side of the frame is the classification stage. Combined with machine learning classification approaches, the features are used to find attackers.

4.2 Feature Extraction Method

In the former section, we give the definitions of popularity profile and popularity distribution of a user, which are the output of data preprocessing stage. For our detection purpose, it is not necessary to operate on all possible values of the popularity distribution [29]. Instead, it is appropriate to consider only a small number of accumulated probability over some intervals. Therefore, we bucket the range of popularity distribution into several intervals to get accumulated probability as features [29], [30]. Then the task becomes how to get the width of each interval to divide the whole range into several intervals. As a toy example, we divide the popularity distribution of users in MovieLens 100K into six intervals when the whole range (1 to 583) is divided according to the width 100 shown in Table 3.

But how to decide the width of the interval remains a question. We notice that the popularity distributions of two types of users are different, and thus we check whether the value of *Mean popularity* can be used as the width to divide the range into intervals. The mean popularity of a user (MPU) refers to the mean value of popularity profile, or the mean value of rated items' popularity in a user profile. MPU is obtained by Formula (3).

$$MPU_u = \frac{\sum_{u=1}^{N_u} d_{u,i}}{N_u} \tag{3}$$

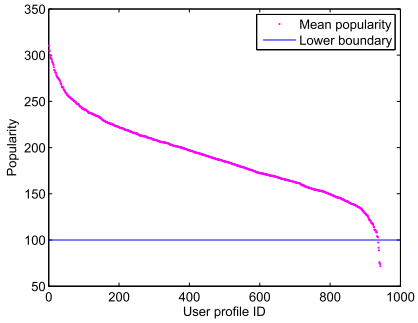
Where N_u is the number of items user u's has rated. $d_{u,i}$ is the popularity of user u's i-th rated item. We show the distribution of normal users' and attackers' MPU in Fig. 6.

Figure 6 (a) shows the distribution of normal users' MPU in MovieLens. It can be found that the values of normal users' MPU are normally larger than a certain value (here is 100). Figure 6 (b) shows the distribution of attackers' MPU with 10% filler size.

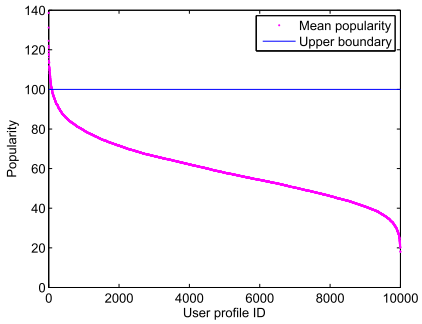
It can be found if attackers select items to rate randomly, their MPU will normally be less than a certain value (here is 100). So here comes our method to decide the width. We set the width of each interval as the lower boundary of all normal users' MPU values. Here we denote these probability values at these k intervals as our features. The general way to decide the width and get features is shown in Table 4.

4.3 Detection of Attackers

After the features are obtained, machine learning classification approaches can be used to train a detector to detect attackers. We treat shilling attack detection as the problem of classification, which is to make a classification between normal users and attacker. Therefore, in this paper, we exploit classification algorithms [31] as the basic model and get the final detection method called Pop-SAD. Pop-SAD is trained on both collected normal user profiles and attacker profiles, and it is able to classify a new user as a normal user or attacker. Our proposed detector is the supervised style, but it can be extended to another kind of machine learning



(a)



(b)

Fig. 6 Distribution of MPU, (a) normal users, (b) attackers

Table 4 Process of obtaining features

Input:
Rating database R, including m normal users and t attackers.
Output:
Interval width W,
Interval number k,
Features matrix F.

1. Get the popularity of each item d_i and find the maximum value of popularity denoted as d_{max} in the rating database.
2. Get the popularity profile, denoted as $PP_u = (d_{u,1}, d_{u,2}, \ldots, d_{u,N_u})$.
3. Get the MPU of each user according to Formula (3). Then find the minimum number of MPU among the m normal users and denoted it as W.
4. Obtain the number of preset interval denoted as K according to d_{max} and W. The calculation is as follows:

$$K = \frac{d_{max}}{W} \tag{4}$$

5. Divide the probability of popularity into K intervals: $(0, 1{\times}W]$, $(1{\times}W, 2{\times}W], \ldots, ((k-1){\times}W, d_{max}]$.
6. Get the accumulated probability at each interval and denote them as a vector represented as $F_u = (f_{u,1}, f_{u,2}, \ldots, f_{u,k})$, which is used as the feature vector too.
7. Form the Feature matrix denoted by $F = [F_1; F_2; \ldots; F_{m+t}]$.

model.

5. Experiments and Discussions

In this section, we present experimental method, results and discussion. First, the experimental method is introduced. Then, three groups of experiment results are shown. The first is conducted on MovieLens 100K dataset to compare the detection accuracy between the proposed detection method (Pop-SAD) and state-of-the-art methods. The second is conducted on the same dataset to see the effectiveness of our proposed feature extraction method. The last is conducted on the Amazon review dataset to show the practical value of the proposed method. Finally, we describe some discussions related to Pop-SAD.

5.1 Experimental Method

- Dataset: We conducted experiments on MovieLens 100K dataset and Amazon review dataset. The MovieLens100K dataset consists of the ratings from 943 users on 1682 items, with a rating frequency not less than 20 for each item [4]. The Amazon review dataset is crawled from Amazon.cn till August 20, 2012, which contains 1205125 reviews written by 645072 reviewers on 136785 products [32]. Each review has 6 attributes: ReviewerID, ProductID, Product Brand, Rating, Date and Review Text. Moreover, 5055 reviewers are labeled in this dataset, with 1822 attackers and 3233 normal users.
- Measures: We used precision, recall and $F1-measure$ to measure the performance. $F1-measure$ is a standard metric with the combination of precision and recall. Precision and recall are the ratio of the predicted true attackers to the predicted attackers and true attackers, respectively. These measures can be calculated as follows.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 - measure = \frac{2 \times Presicion \times Recall}{Presicion + Recall} \tag{7}$$

  Where $\#TP$ is the number of attackers who are correctly identified, $\#TN$ is the number of normal users accurately detected. $\#FN$ is the number of attackers who are avoided being detected and $\#FP$ is the number of normal users who are incorrectly identified as attackers.
- Baselines: For the experiments conducted on the MovieLens 100K dataset, we select Bayes-SAD [22], a supervised classification method; PCA-SAD [17], an unsupervised classification method; Semi-SAD [15], a semi-supervised classification method as the baselines to compare with Pop-SAD in terms of detection accuracy in Sect. 5.2. The analysis of the interval number's effect on detection performance is illustrated in Sect. 5.3. For the experiments conducted on the Amazon review dataset, we choose detection methods using linguistic-based, individual behavioral-based and collusive behavioral-based features as the baselines to compare with our proposed detection method in detecting review attackers in Sect. 5.4. The purpose is to show features extracted from popularity distribution is able to make a distinction between normal users and attackers in different datasets.

5.2 Detection Performance of Different Methods

Pop-SAD uses the features derive from selecting patterns to train a detector while most traditional methods use the features derive from rating features to find attackers. To show the superiority of detecting attackers via selecting patterns analysis, this paper compared Pop-SAD with baselines, i.e. Semi-SAD, Bayes-SAD and PCA-SAD.

Here we choose Naive Bayes to make use of the proposed features to get the proposed method, Pop-SAD. The reasons why Naive Bayes is chosen are two-fold: first, some works aim to detect shilling attacks [15], [17], [22] use Naive Bayes as their basic classifier, and we can compare our method with them; second, the main contribution of our paper is to detect attackers from the perspective of selecting patterns, so choice of a public used method can contribute the detection effect to our proposed features.

We fixed the attack size at 10% and changed the filler size at all levels (5% to 50%) to get the results. We set top-x% fixed to 0.5 in AoP attacks, and ratio between filler size and selected size equals to 1 in bandwagon attacks. Ten target items with different popularity are chosen to generate attack profiles and each time only a target item is used. In order to get unbiased results, we conducted 100 times of experiments for each target item then reported the results according to the average detection accuracy on these ten items.

We used the common test set with 20% of original data for each method at each round of experiment. For the training phase, the rest 80% data were used for training phase of Bayes-SAD, Pop-SAD. For Semi-SAD, we adopt the experiment setup described in [15] and used 20% labeled data and 60% unlabeled data for training. For PCA-SAD, we do not need training set but we reported the result on the common test set. The performance of detecting random attacks, Aop attacks and bandwagon attacks are shown in Figs. 7, 8, and 9. Detection result on hybrid attacks, which is the mixture of these three kinds of attack, with each attack size fixed at 5% is shown in Fig. 10.

It can be found in Figs. 7, 8, and 9 that when detecting single attacks, the accuracy values of Pop-SAD, Semi-SAD, and Bayes-SAD are all acceptable. PCA-SAD fails to work well when detecting Aop attacks, the reason behind maybe that Aop attacks take advantage of popular items, consequently, it causes the misjudgment of normal users. Another aspect is that when the filler size is small, Pop-SAD still works well because it can make use of the popularity

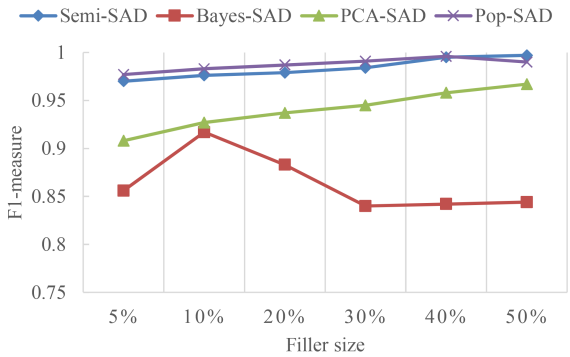


Fig. 7 Performance of different methods on detecting random attacks

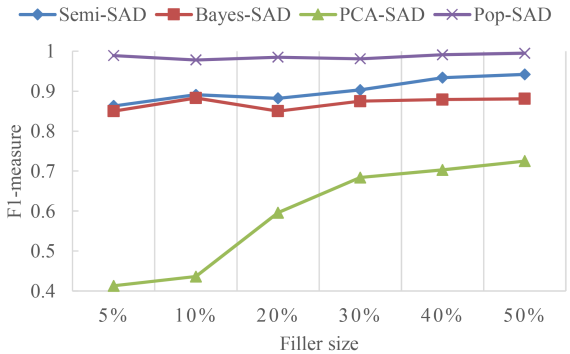


Fig. 8 Performance of different methods on detecting Aop attacks

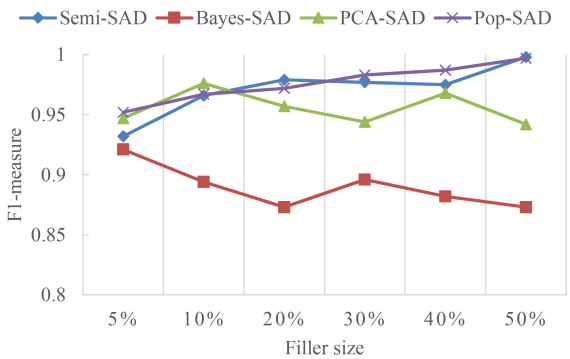


Fig. 9 Performance of different methods on detecting bandwagon attacks

distribution of rated items in user profiles.

From Fig. 10 we can find Pop-SAD outperforms all other three methods in the hybrid attacks. While hybrid attacks make features extracted from ratings hard to locate attackers, features extracted from popularity still work well because the popularity distribution is still powerful to detect attackers.

5.3 Impact of the Number of Intervals on the Detection Performance

We proposed feature extraction method in Sect. 4.2 to get the features. But the number of intervals, which is also the number of features, is a parameter that needs to be decided before. We made k equal to 6 in MovieLens 100K dataset

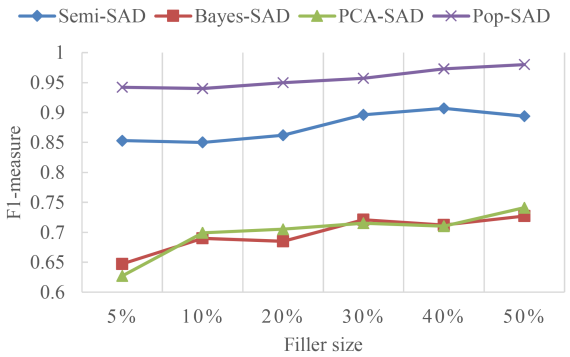


Fig. 10 Performance of different methods on detecting hybrid attacks

Table 5 Impact of the number of intervals (k) on random attack detection

Attack size (%)	Interval number	Filler size (%) 3	6	9	12	15	20	25	30
5	k=3	0.937	0.941	0.959	0.962	0.974	0.987	0.990	0.996
	k=6	0.938	0.943	0.957	0.965	0.974	0.984	0.993	0.996
	k=12	0.913	0.929	0.941	0.935	0.945	0.946	0.946	0.947
10	k=3	0.934	0.940	0.951	0.967	0.977	0.987	0.987	0.995
	k=6	0.968	0.964	0.967	0.970	0.980	0.990	0.994	0.995
	k=12	0.962	0.964	0.970	0.969	0.973	0.974	0.973	0.974
15	k=3	0.974	0.980	0.979	0.979	0.986	0.993	0.997	0.999
	k=6	0.979	0.975	0.978	0.978	0.985	0.992	0.993	0.996
	k=12	0.972	0.976	0.977	0.977	0.980	0.983	0.982	0.982
20	k=3	0.988	0.983	0.983	0.984	0.987	0.995	0.989	0.991
	k=6	0.983	0.984	0.984	0.986	0.985	0.994	0.996	1.000
	k=12	0.983	0.981	0.982	0.983	0.986	0.987	0.987	0.987

Table 6 Impact of the number of intervals (k) on Aop attack detection

Attack size (%)	Interval number	Filler size (%) 3	6	9	12	15	20	25	30
5	k=3	0.684	0.781	0.819	0.819	0.874	0.945	0.974	0.993
	k=6	0.927	0.923	0.934	0.956	0.978	0.988	0.978	0.998
	k=12	0.853	0.949	0.966	0.974	0.976	0.989	0.992	0.984
10	k=3	0.815	0.854	0.876	0.884	0.904	0.959	0.984	0.995
	k=6	0.951	0.954	0.962	0.965	0.978	0.992	0.996	0.998
	k=12	0.901	0.960	0.980	0.991	0.994	0.988	0.995	0.998
15	k=3	0.836	0.881	0.911	0.925	0.923	0.967	0.992	0.997
	k=6	0.964	0.967	0.973	0.974	0.980	0.996	0.998	0.998
	k=12	0.926	0.970	0.988	0.990	0.996	0.993	0.996	0.998
20	k=3	0.869	0.902	0.939	0.938	0.950	0.971	0.993	0.995
	k=6	0.972	0.977	0.979	0.981	0.984	0.995	0.998	0.998
	k=12	0.935	0.969	0.989	0.995	0.996	0.996	0.996	0.998

according to proposed feature extraction method. In this subsection, we added two additional numbers of features, 3 and 12, to show the rationality of our method. Probability in k interval is used as features. The paper demonstrates the comparison of $F1-measure$ for three interval numbers (k) at all levels of attack sizes (5% to 20%) and at all levels of filler sizes (3% to 30%). 10-fold cross-validation was used to get the results. Results on three typical attack models are shown in Tables 5-7.

Table 5 is the effect of hte feature number on the $F1-measure$ of random attacks. From the table, it can be found that the values of interval number have no direct effect on the detection accuracy, but when the value of interval number is too large, the detection performance is not good. We set top-x% fixed to 0.3 in the AoP attacks. The results are shown in Table 6. The value of interval number impacts the detection results a lot, and when the interval number is large, Pop-

Table 7 Impact of the number of intervals (k) on bandwagon attack detection

Attack size (%)	Interval number	Filler size (%) 3	6	9	12	15	20	25	30
5	k=3	0.949	0.981	0.988	0.991	0.993	0.990	0.947	0.990
	k=6	0.956	0.972	0.980	0.974	0.983	0.986	0.988	0.986
	k=12	0.984	0.985	0.984	0.957	0.957	0.948	0.945	0.948
10	k=3	0.965	0.990	0.989	0.998	0.998	0.998	0.974	0.998
	k=6	0.990	0.978	0.988	0.980	0.990	0.992	0.992	0.992
	k=12	0.991	0.991	0.998	0.989	0.982	0.976	0.974	0.974
15	k=3	0.974	0.990	0.994	0.990	0.990	0.990	0.983	0.990
	k=6	0.999	0.987	0.988	0.988	0.993	0.997	0.998	0.995
	k=12	0.993	0.995	0.997	0.996	0.991	0.985	0.981	0.982
20	k=3	0.980	0.997	0.994	0.990	1.000	0.990	0.987	0.990
	k=6	0.999	0.990	0.995	0.990	0.994	0.996	0.997	0.996
	k=12	0.997	0.998	0.997	0.999	0.996	0.990	0.987	0.987

SAD performs not so well. We set the ratio between filler size and selected size equals to 1 in the bandwagon attacks to get the result in Table 7. It can be found that the values of $F1-measure$ first increase then decrease as the values of interval number increase.

From the three tables, it can be concluded our proposed feature extraction method gives a moderate estimation of the number of features. It not only gets a good detection effect but also reduces the computational cost and effectively prevents over-fitting due to its moderate interval number.

5.4 Practical Value of the Proposed Detection Method

To show the practical value of the proposed methods in detecting attackers, we conducted experiments on Amazon review dataset to show the results. The Amazon review dataset is collected from Amazon China[†] by Xu etc [32]. This dataset includes 1205125 reviews written by both normal users and attackers. Our purpose is to detect attackers who wrote the spam reviews. Note that in this dataset, the attackers gave fake ratings while posted spam reviews, so the problem of detection attacker in this dataset equals to the problem of detecting shilling attackers.

In order to apply Pop-SAD to detect spam reviews, we need to extract features from popularity distribution of users according to the feature extraction method introduced in Sect. 4.2. We first analyze the rationality of the proposed feature extraction method, then show the detection accuracy of the proposed detection method.

The maximum popularity of items in this dataset is 3644, and the popularity distribution of items is obtained and drawn in Fig. 11. Here the popularity of an item means the number of reviews it obtained.

It can be found from Fig. 11 that half of the items in this dataset only be reviewed by one customer. Because only a small number of ratings given by users to items can be obtained in this dataset, the popularity distribution of items may be an inaccurate estimation. However, the overall trend of popularity follows power-law distribution, too. We show the MPU distribution of normal users and attackers in Fig. 12.

Figure 12 shows the distribution of 5055 labeled users'

[†]http://www.amazon.cn

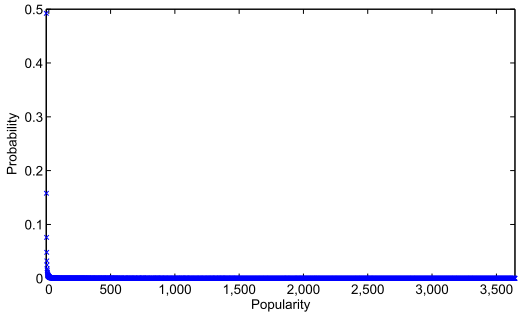


Fig. 11 Property of item popularity distribution in the Amazon review dataset

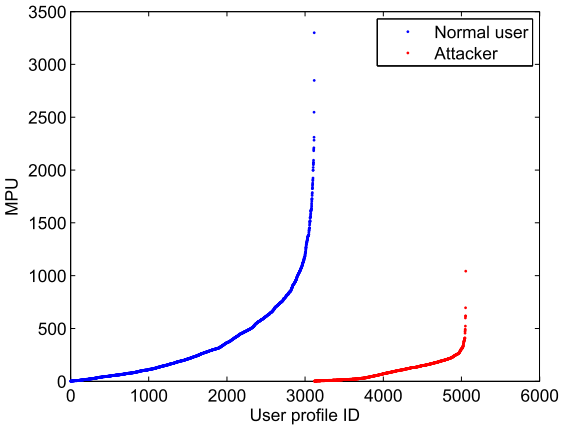


Fig. 12 Distribution of MPU in the Amazon review dataset

MPU in the Amazon review dataset. MPU is the mean popularity of item popularity of a user profile. It can be found that the values of normal users' MPU are normally larger than that of attackers. This means normal users have different selecting behavior with attackers, therefore, it is reasonable to adopt the feature extraction method introduced in Sect. 4.2 to derive features from item popularity in user profiles.

Note that this dataset is only part of the whole rating database of Amazon.cn, so we divide the whole range of popularity distribution according to average value of MPU rather than the minimum value of MPU to get the features. This kind of feature is called popularity-based features in this subsection.

To check the effectiveness of our proposed features, we adopted other three features to make a comparison: linguistic-based features, which are extracted from the text of each user' reviews; individual behavioral-based features, which are extracted from the post behavior of each user; collusive behavioral-based features, which are extracted from the group post behavior of users. More detail about these three features can be found in [32]. KNN and SVM are used to take advantage of features to form detection methods. The reason why KNN and SVM were chosen rather than Naive Bayes is to ensure the proposed features can work well with other machine learning methods. 10-fold cross-validation was used to get the results shown in Table 8.

Table 8 Comparison of different features in detecting shilling attacks

	Feature type	*Precision*	*Recall*	*F1−measure*
KNN	Popularity-based	0.666	0.788	0.722
	Linguistic-based	0.606	0.698	0.648
	Individual behavioral-based	0.658	0.688	0.672
	Collusive behavioral-based	0.795	0.749	0.771
SVM	Popularity-based	0.697	0.806	0.747
	Linguistic-based	0.631	0.624	0.627
	Individual behavioral-based	0.682	0.625	0.652
	Collusive behavioral-based	0.762	0.794	0.777

It can be seen from Table 8 that our proposed features has the largest *recall* rates. Moreover, compared with linguistic-based and individual behavioral-based features, the $F1-measure$ of our proposed features is the best. This indicates that the proposed features are superior to these two traditional features. Though the $F1-measure$ value of the proposed does not outperform that of collusive behavioral-based features, collusive behavioral-based features need to find the group of each user first, which makes it harder to calculate them than our proposed features [32]. All in all, the proposed features are able to detect attackers in the real-life situation, which proves the practical value of our proposed detection method.

5.5 Discussion

5.5.1 Time Cost

In the first two experiments we have shown that detection methods using features extracted from selecting patterns outperformed those using features derived from rating patterns in terms of detection performance. Additionally, we need to check the time cost of computing each kind of feature.

The computation time of the traditional features derived from rating patterns, such as Degsim or RDMA, lies in 1) the calculation of the similarity matrix among users; 2) the calculation of items' mean value of rating and the number of rating times [16]. The similarity matrix computation time is $O(m \times m \times n) = O(m^2 \times n)$, while the time cost of rating related computation is $O(n^2)$, thus O(feature extraction) = $(m^2 \times n)$+$O(n^2) = O((m^2 + n) \times n)$. Here, m is short for the number of users and n for the number of items in the system.

The computation time of the proposed features derived from selecting patterns lies in 1) the statistics of item popularity; 2) getting probability over k intervals. The computation time of item popularity is $O(m \times n)$, while the time cost of probability computation is also $O(m \times n)$. Therefore, O (feature extraction) = $O(m \times n)$ can be obtained.

From the time contrast, we find the computation of proposed features is faster than the traditional features, which is a very large improvement in practice.

5.5.2 Effectiveness Analysis

In the last experiment we have described the practical value of the proposed method, but we need to analyze the effectiveness of our proposed method in a more general way. The purpose of shilling attacks is to manipulate the recommendation results, therefore, attackers need some strategies to select items to rate and give rating values to items, which means selecting patterns and rating patterns respectively. Randomly inserted profiles without attack models will cause no significant impact on the recommender systems [6]. But attackers' knowledge about the recommender systems is limited [13]. Therefore, a user is judged as an attacker only if he adopt some strategic selecting patterns and rating patterns.

Attackers may imitate the real users' selecting and rating patterns and change their rating patterns, hence, features extracted from rating patterns may lose effect as the time pass by. However, no matter what kind of ways the attackers select items to rate, they need to select the target item because that is their purpose of attacks [6], [16]. Finally, attackers need more knowledge to evade our detection, so their cost increases. In the view of the above discussions, the effectiveness of our detection method via selecting patterns analysis remains superior to other methods.

6. Conclusion and Future Works

In this paper, we detected shilling attacks via the analysis of selecting patterns. While item popularity in recommender systems usually follows the power-law distribution, different selecting patterns between normal users and attackers lead to differences in popularity. Thus, we extracted features from the popularity distributions of user profiles. Then, the Naive Bayes algorithm was exploited to use proposed features to get our shilling attack detector Pop-SAD. Experiments on MovieLens 100K dataset and Amazon review dataset show Pop-SAD outperforms detectors built on these features derived from rating patterns in terms of detection performance.

As future work, we plan to combine features derive from rating patterns and selecting patterns to get more accurate detectors. Moreover, other information, such as timestamps and interval of ratings, can also be used to find attackers. Finally, because there are often few labeled but numerous unlabeled users available in practice, detection methods which deal with partial labels of users will be proposed for real-world applications.

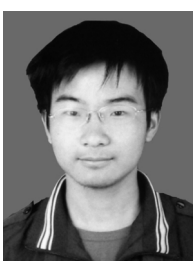
Acknowledgments

This work was supported by the Basic and Advanced Research Projects in Chongqing under the Grant No. cstc2015jcyjA40049, the National Key Basic Research Program of China (973) under the Grant No. 2013CB328903, the National Natural Science Foundation of China under the Grant No. 61502062, the China Postdoctoral Science Foundations under the Grant Nos. 2012M521680 and 2014M560704, and the Fundamental Research Fund for the Central Universities under the Grant No. 106112014CD-

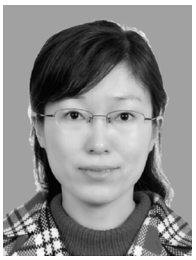
JZR095502.

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," Decision Support Systems, vol.74, pp.12–32, 2015.
[2] K. Truong, F. Ishikawa, and S. Honiden, "Improving accuracy of recommender system by item clustering," IEICE Trans. Inf. & Syst., vol.90, no.9, pp.1363–1373, 2007.
[3] J. Wang and Y. Zhang, "Opportunity model for e-commerce recommendation: right product; right time," Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp.303–312, ACM, 2013.
[4] K. Wei, J. Huang, and S. Fu, "A survey of e-commerce recommender systems," Service Systems and Service Management, 2007 International Conference on, pp.1–5, IEEE, 2007.
[5] B. Bhasker, "Comparative study of collaborative filtering algorithms.," Proc. Int. Conf. Knowledge Discovery and Information Retrieval, Barcelona, Spain, 2012, pp.132–137, 2012.
[6] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: a comprehensive survey," Artificial Intelligence Review, vol.42, no.4, pp.767–799, 2014.
[7] S.K. Lam and J. Riedl, "Shilling recommender systems for fun and profit," Proceedings of the 13th international conference on World Wide Web, pp.393–402, ACM, 2004.
[8] W. Zhou, Y.S. Koh, J. Wen, S. Alam, and G. Dobbie, "Detection of abnormal profiles on group attacks in recommender systems," Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, pp.955–958, ACM, 2014.
[9] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," ACM Transactions on Internet Technology (TOIT), vol.7, no.4, p.23, 2007.
[10] A. Mukherjee, B. Liu, and N. Glance, "Spotting fake reviewer groups in consumer reviews," Proceedings of the 21st international conference on World Wide Web, pp.191–200, ACM, 2012.
[11] H. Zhu, H. Xiong, Y. Ge, and E. Chen, "Ranking fraud detection for mobile apps: A holistic view," Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp.619–628, ACM, 2013.
[12] S. Zhang, A. Chakrabarti, J. Ford, and F. Makedon, "Attack detection in time series for recommender systems," Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp.809–814, ACM, 2006.
[13] R. Burke, B. Mobasher, and R. Bhaumik, "Limited knowledge shilling attacks in collaborative filtering systems," Proceedings of 3rd International Workshop on Intelligent Techniques for Web Personalization (ITWP 2005), 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), pp.17–24, 2005.
[14] B. Mehta, T. Hofmann, and P. Fankhauser, "Lies and propaganda: detecting spam users in collaborative filtering," Proceedings of the 12th international conference on Intelligent user interfaces, pp.14–21, ACM, 2007.
[15] Z. Wu, J. Cao, B. Mao, and Y. Wang, "Semi-sad: applying semi-supervised learning to shilling attack detection," Proceedings of the fifth ACM conference on Recommender systems, pp.289–292, ACM, 2011.
[16] P.-A. Chirita, W. Nejdl, and C. Zamfir, "Preventing shilling attacks in online recommender systems," Proceedings of the 7th annual ACM international workshop on Web information and data management, pp.67–74, ACM, 2005.
[17] B. Mehta and W. Nejdl, "Unsupervised strategies for shilling detection and robust collaborative filtering," User Modeling and User-Adapted Interaction, vol.19, no.1-2, pp.65–97, 2009.
[18] F. Zhang and Q. Zhou, "Hht–svm: An online method for detecting profile injection attacks in collaborative recommender systems," Knowledge-Based Systems, vol.65, pp.96–105, 2014.
[19] C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik, "Detection of obfuscated attacks in collaborative recommender systems," Proceedings of the ECAI'06 Workshop on Recommender Systems, Citeseer, 2006.
[20] Z. Wu, J. Wu, J. Cao, and D. Tao, "Hysad: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation," Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp.985–993, ACM, 2012.
[21] N. Hurley, Z. Cheng, and M. Zhang, "Statistical attack detection," Proceedings of the third ACM conference on Recommender systems, pp.149–156, ACM, 2009.
[22] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp.542–547, ACM, 2006.
[23] Z. Zhang and S.R. Kulkarni, "Graph-based detection of shilling attacks in recommender systems," 2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pp.1–6, IEEE, 2013.
[24] Z. Zaier, R. Godin, and L. Faucher, "Evaluating recommender systems," 2008 International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution, AXMEDIS'08, pp.211–217, IEEE, 2008.
[25] W. Rong, B. Peng, Y. Ouyang, K. Liu, and Z. Xiong, "Collaborative personal profiling for web service ranking and recommendation," Information Systems Frontiers, vol.17, no.6, pp.1265–1282, 2015.
[26] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," Proceedings of the 2008 ACM conference on Recommender systems, pp.11–18, ACM, 2008.
[27] T. Zhou, H.A.T. Kiet, B.J. Kim, B.-H. Wang, and P. Holme, "Role of activity in human dynamics," EPL (Europhysics Letters), vol.82, no.2, p.28002, 2008.
[28] J.R.M. Hosking and J.R. Wallis, "Parameter and quantile estimation for the generalized pareto distribution," Technometrics, vol.29, no.3, pp.339–349, 1987.
[29] C. Krügel, T. Toth, and E. Kirda, "Service specific anomaly detection for network intrusion detection," Proceedings of the 2002 ACM symposium on Applied computing, pp.201–208, ACM, 2002.
[30] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," ACM SIGCOMM Computer Communication Review, vol.35, no.4, pp.217–228, ACM, 2005.
[31] N.B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," Proceedings of the 2004 ACM symposium on Applied computing, pp.420–424, ACM, 2004.
[32] C. Xu, J. Zhang, K. Chang, and C. Long, "Uncovering collusive spammers in chinese review websites," Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp.979–988, ACM, 2013.



Wentao Li received the B.A. degree in educational technology from Nanchang Hangkong University, China in 2013. Currently, He is a M.S. candidate at Chongqing University, China. His research interests include anomaly detection, recommender system and graph mining.



Min Gao received the M.S. and Ph.D. degrees in computer science from Chongqing University, China in 2005 and 2010 respectively. She is an associate professor at the School of Software Engineering, Chongqing University. She was a visiting researcher at the School of Business, University of Reading, UK. Her research interests include recommender system, service computing, and data mining. She has published more than 30 refereed journal and conference papers in these areas. She has grants from the National Natural Science Foundation of China, the China Postdoctoral Science Foundation, and the China Fundamental Research Funds for the Central Universities.

Hua Li received the M.S. degree in computer science from Chongqing University, China in 1988. She is an associate professor at the College of Computer Science, Chongqing University. Her research interests include computer network, recommender system, and big data mining. She has published more than 50 refereed journal and conference papers in these areas.

Jun Zeng received his B.S. and M.S. degrees in software engineering from Chongqing University, China in 2007 and 2010. He received the Ph.D. degree in Advanced Information Technology of Kyushu University, Japan in 2013. Since 2013, he has been a lecturer in the School of Software Engineering of Chongqing University. His research interests include artificial intelligence, data mining, and search engine.

Qingyu Xiong is the dean of the School of Software Engineering, Chongqing University, China. He received the B.S. and M.S. degrees from the School of Automation, Chongqing University, China in 1986 and 1991 respectively, and the Ph.D. degree from Kyushu University, Japan in 2002. His research interests include neural networks and their applications. He has published more than 100 journal and conference papers in these areas. Moreover, he has more than 20 research and applied grants.

Sachio Hirokawa received the B.S. and M.S. degree in Mathematics and Ph.D. degree in Interdisciplinary Graduate School of Engineering Sciences from Kyushu University, Japan in 1977, 1979, and 1992. Since 1997, he has been a professor in the research institute for information technology of Kyushu University, Japan. His research interests include artificial intelligence, search engine, text mining, and computational logic.